DE FACTS II Software Delivery Methodology
Hybrid-Agile
March 2016
Content

Hybrid-Agile Framework
Redefining Roles
Roles & Events – Mapping Responsibilities
Sample Sprint Structure
Testing
Sprint Entry & Exit
Appendix
Our Need

Redefining the FACTS II methodology

DSCYF and Deloitte see the opportunity to modify the approach of FACTS II delivery methods and incorporate lessons learned from the original FACTS II effort and other Deloitte efforts.

The proposed approach is to align a Hybrid-Agile approach by incorporating Agile practices while keeping within the parameters of typical fixed scope projects.

Hybrid-Agile approach blends Agile concepts with the predictability of defined scope, which is delivered iteratively and with transparency.
FACTS II - Hybrid-Agile Approach

**Hybrid-Agile** is the application of adaptive, Agile concepts and techniques in a traditional, predictive project. Below is the ideal state for our effort

Factors to consider when determining what aspects of Agile to apply to a project
Hybrid-Agile Framework – Key Characteristics

Our Hybrid-Agile process is a modular, flexible approach which applies Agile techniques to iterative development and test cycles.

- Organizing work into feature or process focused sprints within a phase.
- Use of a Product Backlog that continuously defines and refines sprint plans.
- Defining fixed scope deliverable or tasks in sprint plans with estimated time and resources.
- Frequent demonstrations of software to the client and gathering of feedback.
- Organizing work into prioritized feature focused sprints
- Tracking effort during sprints using a burn down chart to provide visibility and manage trends.
Hybrid-Agile benefits

Overall project is executed on a priority basis, taking into account constraints and dependencies.

Consistent reviews at the end of each sprint mean any issues will surface early and can be addressed immediately.

A process focus brings all groups together and removes silos.

Stakeholders will be involved in every sprint and tangible progress will be reviewed regularly, supporting continuous feedback between development and the business.

Focusing on shorter sprints with targeted functionality allows Teams to develop prototypes to confirm requirements.

Benefits

- Ability to accelerate delivery of high priority features
- Enhanced Team collaboration
- Increased transparency and feedback
- Issues are identified early
- Ability to rapidly prototype solutions
Success factors

**Highly engaged stakeholders**
Business representatives and other key stakeholders must be available to the development team to answer questions and provide feedback on a timely and frequent basis.

**Persistent, continuous focus on requirements**
Requirements priority may be driven by business value, technical dependencies, process dependencies, or impact to other requirements.

New requirements are likely to emerge during sprint reviews and should be prioritized against the exiting scope of work, and are subject to change control.

**Fixed sprint scope and time-boxed iterations**
Once a sprint has started, the scope is frozen. Sprints are strictly time-boxed, and any unfinished work must be reprioritized against the future sprint plans.

**Consistency**
Using the Hybrid-Agile approach in daily activities requires a disciplined approach to consistently report burn down and manage to sprint plan and goals.
Hybrid-Agile Framework
Application of Agile concepts

Agile concepts typically used in Hybrid-Agile approach include:

- **Requirements** are “force-ranked” so that each has a **unique priority value** (1, 2, 3, etc.), and priorities may be adjusted as new information becomes available or new requirements emerge from sprint reviews.

- **Continuous Requirements Prioritization**

- **Team Capacity & Velocity**

- **Hybrid-Agile Concepts**

- **Daily Stand-Ups**

- **Sprint Burn Down**

- **Ongoing Planning**

- **Sprint Reviews & Retrospectives**

- **Team capacity** = the **amount of hours available** per sprint for work on the sprint deliverables.

- **Team velocity** = number of **units of work completed** in a sprint.

- **15 minute time-boxed meetings** to review work accomplished the day before, work planned for the current day, and any impediments facing the team.

- **A visual representation** which shows the teams’ overall progress against the sprint goals.

- **Continuous planning** supports **flexibility** in incorporating new requirements or changes in priorities (subject to change control processes).

- **Sprint reviews** focus on demonstrating work completed during the sprint and on gathering feedback from stakeholders.

- **Sprint retrospectives** are used to identify **lessons learned** and improvements for future sprints.
FACTS II Hybrid-Agile Framework

Design and development work is conducted in **Sprints**, with some testing conducted concurrently. An integrated end-to-end test phase is conducted at the conclusion of sprints and prior to a **Release**.

---

**Plan**
- Develop Master Plan
- Develop Work Plan

**Analyze**
- Sprint 0
  - Prioritize Business Requirements
  - Prioritized Processes (for package implementation)
  - High-level Design
  - Develop Sprint Plans

---

**Design & Build**
- Design, Build, and Unit/System Test Sprints
  - Number of sprints determined by project
  - Sprint plans developed iteratively

**Test**
- Initial Integration & Regression Test Sprints
  - Integration and regression testing starts earlier
  - Testing is executed as code is ready

**Deliver**
- Incremental UAT Sprints
  - Incremental UAT Sprints conducted with completion of Program Area User Stories
- Data Migration, Conversion (where appropriate)

---

**Technology/Infrastructure**

**Project & Quality Management**

**Organizational Change Management**

---

***Note: there will be multiple Incremental UAT Sprints (up to 4) to be determined in Sprint 0. Only 1 is depicted in diagram.***

---

Copyright © 2016 Deloitte Development LLC. All rights reserved.
Discovery - Planning and Analysis

During the planning and analysis, the foundation for the project is cemented. An overarching master plan is created, themes and epics are mapped on a roadmap and the product backlog is created with initial user stories that are prioritized.

As part of sprint 0 the definition of ready and definition of done, must be established and the user stories for the first two sprints built out to a level of detail that can be accepted as ready. Foundation architecture strategies, testing, training and communication strategy's are established for the project.
Design & Build - Sprint

A sprint represents a four week (working days) period in which the team works on User Stories from the Sprint Backlog a subset of Product Backlog with the goal of producing working software.

Product Backlog

The product backlog contains all of the User Stories in scope for the project.

Sprint 1 Sprint Plan

The Sprint’s features or capabilities are demonstrated and evaluated at the end of each Sprint by End Users and Stakeholders to determine if a User Story meets the defined Acceptance Criteria for that Story.

Sprint Readiness

- All User Stories for the Sprint have met Definition of Ready:
  - Acceptance Criteria
  - Story Point Estimates
  - High-level design
  - Capacity estimation

A subset of User Stories are selected to be worked on during a Sprint cycle with the goal of producing a working feature or capability of the software.
# Anatomy of a Sprint

<table>
<thead>
<tr>
<th>Mon</th>
<th>Tue</th>
<th>Wed</th>
<th>Thu</th>
<th>Fri</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sprint Planning</td>
<td></td>
<td></td>
<td></td>
<td>Standup</td>
</tr>
<tr>
<td>Define/Assign</td>
<td></td>
<td></td>
<td></td>
<td>Standup</td>
</tr>
<tr>
<td>Development Tasks</td>
<td></td>
<td></td>
<td></td>
<td>Standup</td>
</tr>
<tr>
<td>Define/Assign</td>
<td></td>
<td></td>
<td></td>
<td>Standup</td>
</tr>
<tr>
<td>Testing Tasks</td>
<td></td>
<td></td>
<td></td>
<td>Standup</td>
</tr>
</tbody>
</table>

**Development**

**Week 2**

<table>
<thead>
<tr>
<th>Standup</th>
<th></th>
<th></th>
<th>Standup</th>
<th>Standup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Write System Tests</td>
<td></td>
<td></td>
<td></td>
<td>Standup</td>
</tr>
<tr>
<td>Write Integration Tests</td>
<td></td>
<td></td>
<td></td>
<td>Standup</td>
</tr>
</tbody>
</table>

**Week 3**

<table>
<thead>
<tr>
<th>Standup</th>
<th></th>
<th></th>
<th>Standup</th>
<th>Standup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Write Integration Tests</td>
<td></td>
<td></td>
<td></td>
<td>Standup</td>
</tr>
<tr>
<td>Development</td>
<td>Defects Remediation</td>
<td></td>
<td>Execute System Testing</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Design and Refinement for the following Sprint</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Review Sprint User Stories</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Standup</td>
<td>Standup</td>
<td></td>
<td>Standup</td>
<td></td>
</tr>
<tr>
<td>Defects Remediation</td>
<td>Execute System Testing</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Review Sprint User Stories</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Week 4**

<table>
<thead>
<tr>
<th>Standup</th>
<th>Standup</th>
<th>Standup</th>
<th>Sprint Review Session</th>
<th>Sprint Retrospective</th>
</tr>
</thead>
<tbody>
<tr>
<td>Defects Remediation</td>
<td>Execute System Testing</td>
<td></td>
<td>Product Backlog Refinement</td>
<td></td>
</tr>
<tr>
<td>Review Sprint User Stories</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Design and Refinement for the following Sprint**

*DSCYF Activity*  *Deloitte Activity*  *Joint Activity*
Design & Build – Initial Integration Testing

After User Stories are accepted (have met acceptance criteria) **at the end of a Sprint**, they will migrate to an Integration environment for initial integration testing.

The accepted Sprint’s features or capabilities move into the Initial Integration Testing Sprint.

**Initial Integration Testing begins** on all accepted User Stories.

User Stories have passed all **integration tests** and are now fit in with other system functionalities.
The Release – Incremental UAT

A release is a grouping of User Stories that represents a system functionality (e.g. Program Involvement). A Release is ready for Incremental UAT once all User Stories in that Release are considered “Done” and have passed Integration Testing.

The product backlog contains all of the User Stories in scope for the project.

User Stories will be grouped into system functionality categories called Releases. (E.g. Program Involvement)
A Release is “Done” when all of its User Stories are considered “Done.”

DSCYF will execute end-to-end Incremental UAT testing of the Release items to validate that the system meets the business needs for that Release.
Redefining Roles
The Importance of Sprint Roles

The FACTS II methodology emphasizes the importance of project roles — Product Owner, Scrum Master, and Team — and includes steering and oversight from the executive team.

### Executive Product Owner
- **Client voice** and accountable for product success
- Attends Daily Stand-Up Meetings
- Works with Scrum Master to clear impediments
- Defines high-level product features and priority of user stories working with stakeholders
- Manages the **Product Backlog**
- Ensures user stories and acceptance criteria are what end-users need
- Attends Sprint Reviews
- Ultimately decides if every **User Story** in the **Sprint Review** meets the **Acceptance Criteria**

### DSCYF Project Management

### Deloitte Project Management

### Executive Scrum Master
- **Servant Leader** — enables close cooperation across all roles/functions
- Facilitates daily 15 minute team meeting (**Daily Stand-up**)
- Maintains **Sprint Burn Down Charts**
- Manages **Impediments** identified by the team
- Protects team from distractions

### Product Owner (DSCYF)
- Cross-functional, self-managing, 5-9 members
- Performs the tasks of delivering the product
- Team defines own tasks and assignments
- Maintains own **Sprint Backlog**
- Conducts **Sprint Review**

### Scrum Master (Deloitte)

### Team** (DSCYF + Deloitte)

### SMEs

### Primary Client Stakeholder

### Stakeholders

** team roles defined in following slides
Roles & Events
Mapping responsibilities
Phases, Events & Activities Require Consistent Involvement

**Sprint 0**

**PMO Deliverables**

**Define Themes, Epics and User Stories Product Backlog.**

**Develop Master Plan including Themes, User Stories, and Releases**

**Sprint Cycle**

- **Conduct Daily Stand-up to adjust to achieve the completion of the Sprint Backlog.**
- **Team works closely with Product Owner to complete User Stories in Sprint Backlog.**
- **Conduct Sprint Planning to assign User Stories from Product Backlog to Sprint Backlog. Finalize Definition of Done for User Stories**

**Conduct Sprint Review to demonstrate working software to Product Owner and Stakeholders**

**Sprint Cycle**

- Approved User Story
- **Rejected User Story moved to Product Backlog**

**UAT Release**

- Working Software deployed to UAT Testing Environment

**Sprint ends with Sprint Retrospective to discuss lessons learned**

Copyright © 2016 Deloitte Development LLC. All rights reserved.
Breakdown of Product Backlog Items

Overview

- **Themes**
  - A Theme is a top-level objective that may span projects and products. Themes may be broken down into sub-themes, which are more likely to be product-specific. At its most granular form, a Theme may be an Epic.
  - **Example**: Personalize user experience

- **Epics**
  - An Agile Epic is a group of related User Stories, representing a feature. You would be unlikely to introduce an Epic into a sprint without first breaking it down into its component User Stories so as to reduce uncertainty.
  - **Example**: Allow users to create user profiles

- **User Story**
  - A User Story is a description of desired functionality told from the perspective of the user.
  - **Example**: As a registered user, I want to reset my password so that I can login into the site if I forget my password

- **Task**
  - Tasks are individual activities that are required for a User Story to be “done”.
  - **Example**: Author acceptance test for User
What is a Product Backlog?

- An ordered list of requirements
- Contains features, functions, requirements, enhancements, defects, etc.
- Dynamic and constantly changing
- Ordered by the Product Owner

Epic 1: Allow Users to Create User Profiles

User Story 1
User Story 2
User Story 3
User Story 4
User Story 5
User Story 6
User Story 7
User Story 8

Business Value Priority
Discovery – Product Backlog

At the beginning of the project, the **Product Owner** works with the primary client stakeholder to identify Epics and User Stories which are grouped together to create **Themes** for the **Product Backlog**.

**Product Backlog**

**Theme 1: Personalize user experience**

- **Epic 1**
  
  *Allow users to make FACET Intake*

- **Epic 2**
  
  *Allow users to make PSSF Intake*

**Theme 2: Increase traceability of clients**

- **Epic 3**
  
  *Allow users to have clients in multi-division program involvements*

- **Epic 4**
  
  *Allow providers to be involved in multi-divisional services*
Creating Well Defined User Stories

A user story:

- Is a basic unit of work representing some business value that can be delivered within a sprint
- Includes enough detail to enable the project team to make planning decisions
- Consists of one or more sentences used to describe user needs and act as a placeholder for a conversation
- Captures the “who”, “what”, and “why” in a simple and concise way
- Is used to describe requirements for different users (personas)

As
The system user or the persona who will be using this story

I want to
Achieve a goal as a result of using the system

So that
Tangible benefits that will be realized after using the system
Sample User Story Template

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Story Title</td>
</tr>
<tr>
<td>2</td>
<td>MVP #</td>
</tr>
<tr>
<td>3</td>
<td>Users/Personas and Systems Involved</td>
</tr>
<tr>
<td>4</td>
<td>Story Summary</td>
</tr>
<tr>
<td>5</td>
<td>Acceptance Criteria 1</td>
</tr>
<tr>
<td>6</td>
<td>Free Drawing</td>
</tr>
<tr>
<td>7</td>
<td>Supported Scenarios</td>
</tr>
<tr>
<td>8</td>
<td>Assumptions / Risks / Unknowns</td>
</tr>
</tbody>
</table>

Example:

```
As a... I want... So that...

Acceptance Criteria 1
Acceptance Criteria 2
Acceptance Criteria 3

Given
AND
When
AND
Then
AND

Given
AND
When
AND
Then
AND

Assumptions / Risks / Unknowns
```

- **Required**
- **Optional**
## Sample User Story Template Guidelines

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Enter the Story Title, MVP, and Story ID, and refine the Story Title as the User Story evolves to reflect what it is</td>
</tr>
<tr>
<td>2</td>
<td>Enter the estimate in story points to deliver the User Story as decided by the team in an estimation session just in time for the story to enter the sprint</td>
</tr>
<tr>
<td>3</td>
<td>Put the User Story into a high level structure soon after the story has been identified (through Story Mapping). This should happen early in a story’s life during Product Backlog Refinement (PBR) but should be refined as the story becomes more definite and better understood</td>
</tr>
<tr>
<td>4</td>
<td>Identify the Users relevant to the User Story including any Personas created by the team and Systems involved in the User Story development. This can happen early in a story’s life but should be refined as the user group becomes more defined and the system context better understood</td>
</tr>
<tr>
<td>5</td>
<td>Define Acceptance Criteria for the User Story while trying to keep the story as simple as possible (treat each criteria as additional scope and therefore potential complexity and increased time to deliver). This should happen during PBR and these criteria should be refined as the story becomes more definite and better understood</td>
</tr>
<tr>
<td>6</td>
<td><strong>Optional:</strong> Create a visual representation to help convey what the User Story is about; it could be a UI mockup, a system interaction diagram, a decision tree, etc.. This can happen throughout a story’s life as the team tries to think through the requirement and uses visualization techniques to do so</td>
</tr>
<tr>
<td>7</td>
<td>Elaborate the scenarios that the User Story needs to support in order to start fleshing out the requirement and giving the team the detail needed to estimate. This would include any ‘happy path’ and ‘exception’ scenarios and should happen (and be refined) during PBR and be completed just in time for estimation</td>
</tr>
<tr>
<td>8</td>
<td><strong>Optional:</strong> Track any assumptions, risks or unknowns that come up in relation to the User Story that the team needs to validate, address, or resolve. This can happen through a story’s life as the team explores the story</td>
</tr>
</tbody>
</table>
Sprint Cycle – Sprint Planning & Sprint Backlog

At the beginning of every Sprint, the Scrum Master facilitates a Sprint Planning meeting with the team and Product Owner to validate the User Stories, estimate any stories not previously estimated and assign to the Sprint Backlog based on size and priority. This allows the teams to accurately adjust based on capacity and velocity. Executive Scrum Master provides support if needed.

Team responsibilities:

- **Self-assigns & commits to tasks**
- Breaks down each user story into tasks
- Decides when capacity met for Sprint and Sprint Backlog is filled
Sprint Cycle – Daily Stand-up

Everyday during the Sprint, the Scrum Master facilitates the Daily Stand-up with the Team and Product Owner to discuss and assess Sprint progress. For projects containing multiple scrum teams there will also be a scrum or scrum meeting.

1. What stories or tasks were finished yesterday?

2. What stories or tasks are you working on today?

3. Can you help us clear these items that are blocking us?
Sprint Cycle – Sprint Review

At the end of the sprint, the team demonstrates working software to the Stakeholders, Product Owner, and SMEs and gathers feedback. The Product Owner decides if each User Story in the Sprint Backlog meets the Acceptance Criteria for the given User Story.

Executive Product Owner serves as tie-breaker on special situations.

Team demos working software to Product Owner.

We like X but can you add Y?

This feature doesn’t seem to be working correctly.
Sprint Cycle – Sprint Retrospective TEAM ONLY

To conclude the sprint, the **Scrum Master** facilitates a **Sprint Retrospective** with the **team** to discuss lessons learned and identify best practices. This is an internal team event that facilitates open and honest progressive growth.

**Scrum Master**

**What did we do well?**

**What didn’t go so well?**

**What should we do differently next time?**

**Team**

- *Fewer stakeholders should be invited to the Sprint Review.*
- *We need to better define our user stories.*
- *I think we need to have 4 week sprints.*
- *I think Daily Stand-up should be in the morning.*
Sample Sprint Structure
A Staggered Sprint Schedule

Staggering sprints distributes the client’s availability to work with different functional teams during each sprint. This way, the client can focus on sprint activities for different functionalities one week at a time.
Release Plan Grooming – 3 sprints in advance

Release Planning will be completed during Sprint 0. Each Sprint Team will do Release Plan Grooming at the end of every sprint in order to defer pending User Stories to future sprints.

<table>
<thead>
<tr>
<th>Sprint 0</th>
<th>Month 1</th>
<th>Month 2</th>
<th>Month 3</th>
<th>Month 4</th>
<th>Month n</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 2 3 4</td>
<td>1 2 3 4</td>
<td>1 2 3 4</td>
<td>1 2 3 4</td>
<td>1 2 3 4</td>
</tr>
</tbody>
</table>

Team 1

Product Backlog

Release Plan Grooming will help re-plan pending User Stories and place them in future sprints.
Testing
### Facts II Hybrid-Agile Test Framework

#### Plan
- Develop Master Plan
- Develop Work Plan

#### Analyze
- Sprint 0
  - Prioritize Business Requirements
  - Prioritized Processes
  - High-level Design
  - Develop Sprint Plans

#### Design & Build
- **Design, Build, and Unit/System Test Sprints**
  - Number of sprints determined by project
  - Sprint plans developed iteratively

#### Test
- Integration, Parallel Performance & Regression Test
- Initial Integration & Regression Test Sprints
  - Integration and regression testing starts earlier
  - Testing is executed as code is ready

#### Deliver
- User Acceptance Test
- Deployment
- Support

#### Data Migration, Conversion, Load Testing, Performance Testing

#### Technology/Infrastructure

#### Project & Quality Management

#### Organizational Change Management
FACTS II Hybrid-Agile Test Framework
Unit, System, Integration, and Incremental UAT testing.

**Design & Build**

Design, Build, and Unit/System Test Sprints

- **Unit Test** – Run manual and automated unit tests several times a day on affected code to cut down on defects at the local environment.

- **System Test** – Run System tests to ensure the system functionality meets the business requirements, including alternate flows, and is ready for the sprint review.

- **Initial Integration Test** – We will run initial integration tests to validate the end-to-end flow of business requirements within the use cases, including alternate flows.
FACTS II Hybrid-Agile Test Framework

Unit, System, Integration, and Incremental UAT testing.

**Design & Build**
- **Design, Build, and Unit/System Test Sprints**

**Initial Integration & Regression Test Sprints**

**Incremental UAT Sprints**

**Unit Test** – Run manual and automated unit tests several times a day on affected code to cut down on defects at the local environment.

**System Test** – Run System tests to ensure the system functionality meets the business requirements, including alternate flows, and is ready for the sprint review.

**Integration Test** – We will run integration tests to validate the end-to-end flow of business requirements within the use cases, including alternate flows.

**Incremental UAT** – DSCYF will execute the UAT test cases designed by the Product Owner.
**Testing Approach – Test Scripts**

Deloitte and DSCYF will each write and compare their corresponding test scripts to ensure proper testing of each requirement.

<table>
<thead>
<tr>
<th>Deloitte</th>
<th>Sprint Team</th>
<th>Integration Testing</th>
<th>Incremental UAT Testing</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Developer</strong></td>
<td>Unit</td>
<td>Unit Test</td>
<td></td>
</tr>
<tr>
<td><strong>Sprint Tester</strong></td>
<td>System</td>
<td>System Test</td>
<td></td>
</tr>
<tr>
<td><strong>Integration Tester</strong></td>
<td>Integration Test</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**DSCYF**

<table>
<thead>
<tr>
<th>Product Owner</th>
<th>Incremental UAT</th>
</tr>
</thead>
</table>

- **Write Test Script**
- **Execute Test Script**

**Sprint Review (Definition of Done)**
Testing Approach – Test Scripts

Deloitte and DSCYF will each write and compare their corresponding test scripts to ensure proper testing of each User Story.
Testing Approach – Defect Lifecycle Management within a sprint

1. Defect raised
2. New CR?
   - Yes: Approved?
     - Yes: Product Backlog
     - No: Defect Valid?
   - No: Triage
3. Defect Valid?
   - No: Close/Reject/Duplicate/Obsolete
   - Yes: Accepted in Sprint?
     - Yes: Test Lead changes status to “In Development”
       - Defect status changes to “Failed Test” (re-open), User Story goes back to “In Development”
       - Test Lead/Developer input detail into tracker including time estimate to fix and functional impact
       - Defect prioritization
         - Test Lead and Design Lead prioritize redevelopment activity
         - Defect development
           - Development team work to fix the requirement
           - Req re-testing
             - Test Lead changes the Defect status to “Ready for Retest”
             - Same defect?
               - No: Fixed?
                 - Yes: Closed
                   - Tester changes Defect status to “Closed”
               - Yes: Defect clarification
                 - Test Lead/Developer input detail into tracker including time estimate to fix and functional impact
4. Rejected
   - No: Scrum Master/Executive Product Owner?
Testing Approach – Defect Lifecycle Management in Integration Testing and Incremental UAT

- **Triage**
  - **Defect raised**
    - **New CR?**
      - Yes: **Rejected**
        - Scrum Master/Executive Product Owner?
          - Could be: Training, User Error, Duplicate Defect
      - No: **Approved?**
        - Yes: **Product Backlog**
        - No: **Defect Valid?**
          - No: **Close/Reject/Duplicate/Obsolete**
          - Yes: **Blocker?**
            - Yes: **Defect clarification**
              - Test Lead/Developer input detail into tracker including time estimate to fix and functional impact
            - No: **Defect prioritization**
              - Test Lead and Design Lead prioritize redevelopment activity
            - **Defect development**
              - Development team work to fix the requirement
            - **Req re-testing**
              - Test Lead changes Defect status to “Ready for Retest”
    - No: **Defect status changes to “Failed Test” (re-open), User Story goes back to “In Development”**
  - **Same defect?**
    - No: **Fixed?**
      - Yes: **Closed**
        - Tester changes Defect status to “Closed”

---

Copyright © 2015 Deloitte Development LLC. All rights reserved.
Defect Severity –

Defect Severity is determined by the person that creates a defect and measures the degree of negative impact on the quality of software. Severity is applied to integration and UAT testing.

- **S4**
  - **Definition:** Cosmetic or trivial errors.
  - **Examples:** visual changes (color), screen wording (internal)

- **S3**
  - **Definition:** Functionality missing on a screen. Workaround available.
  - **Examples:** A simple business rule or validation missing on a screen.
    - “Error Message is not in sync with design spec.”

- **S2**
  - **Definition:** Incorrect functionality and no acceptable workaround available. Stops Testing.
  - **Examples:** “Failed test case because drop-down does not display values, therefore cannot continue.”

- **S1**
  - **Definition:** Any fatal error preventing the user to complete a transaction. Stops Testing.
  - **Examples:** “Page crash on click of ‘save’ (ContractDocuments.aspx)”
**Defect Priority**

Defect Priority refers to the importance of a defect from a business value perspective as evaluated by the development team.

<table>
<thead>
<tr>
<th>Level</th>
<th>Definition</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 – Urgent</td>
<td>• Defect affects the application severely</td>
<td>• Fix immediately&lt;br&gt;• Must be fixed before sprint cycle is complete</td>
</tr>
<tr>
<td>2 – Medium</td>
<td>• The defect is not urgent yet still impacts the application's functionality</td>
<td>• Fix in normal course of activities&lt;br&gt;• Should be fixed during sprint cycle but could be deferred to future sprint</td>
</tr>
<tr>
<td>3 – Low</td>
<td>• Defect is irritant but does not affect key application functionality</td>
<td>• Can wait until after more serious defects have been fixed</td>
</tr>
</tbody>
</table>

**Defect Categorization Example Matrix**

<table>
<thead>
<tr>
<th>Severity</th>
<th>Priority</th>
<th>Example Defects</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>• Any spelling mistake within a paragraph or report (not in title, heading, logo, etc)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Spelling mistake on application title, cover page, logo</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Application crash preventing a workflow happening on a screen that is rarely used</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Application crashes on frequently used screens</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Defect that severely corrupts environment data</td>
</tr>
</tbody>
</table>
Test Data Management

The project will follow a Test Data Management guideline to set rules for Governance, Test Data Sourcing, and Data Refreshes.

<table>
<thead>
<tr>
<th>Governance</th>
<th>Sprint Environment</th>
<th>Integration Testing Environment</th>
<th>Incremental UAT Environment</th>
<th>UAT Environment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shared Ownership between Deloitte Development and Testing team</td>
<td>Deloitte Testing Team</td>
<td>DSCYF</td>
<td>DSCYF</td>
<td></td>
</tr>
</tbody>
</table>

| Test Data Source                  | Manually keyed Data as per test case requirements | Data from Excel sheets or DB Scripts and manual as per the test case requirements | Data from Excel Sheets or DB Scripts as per the test case requirements | Converted Data |

<table>
<thead>
<tr>
<th>Refresh Schedule</th>
<th>On Demand</th>
<th>On Demand</th>
<th>On Demand</th>
<th>To be agreed in Sprint 0</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Refresh Schedule Details</th>
<th>1. After defect fixes that caused data corruption</th>
<th>1. After defect fixes that caused data corruption</th>
<th>1. After defect fixes that caused data corruption</th>
<th>1. After defect fixes that caused data corruption</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2. Before conducting sprint demos</td>
<td>2. When a new table is added to the schema</td>
<td>2. After client testing team is done testing and asks for a refresh when required</td>
<td>2. Refresh data after agreed period deterrent in conversion approach</td>
</tr>
<tr>
<td></td>
<td>3. When a new table is added to the schema</td>
<td>3. After testing team is finished testing and asks for a refresh</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Sprint Entry & Exit
User Story Lifecycle - Sprint Entry Criteria

**Definition of Ready** – the team decides if a user story has met the agreed criteria prior to being accepted into a sprint.

1. The user story is clear to all team members.
2. The user story is descriptive enough that team members will be able to identify the tasks for that story.
3. The User Story passes the INVEST check (Independent, Negotiable, Valuable, Estimate, Small, Testable).
4. Acceptance Criteria is defined, reviewed, and approved by PO/Tech lead or team.
5. User story has been sized by the team.
6. User story can be completed within the sprint according to DoD
7. Inter team and resource dependencies have been identified and resolved.
8. Team knows how to demo the user story.
9. All outstanding questions have been answered with respect to the user story.
10. User story is entered into the mutually agreed AGILE TOOL and all mandatory fields are populated.

In order to accept the user story in a sprint, it must meet the INVEST criteria.
User Story Lifecycle - Sprint Exit Criteria

Definition of Done – defined during sprint 0 this serves as the criteria for the product owner to consider an item “done.” It is recommended to follow the SMART criteria to decide the criteria for “done.”

1. Configuration and code development for user story has been completed
2. Manual Unit/Feature test cases developed and documented
3. Manual Unit/Feature tests 100% executed and all bugs have been fixed – 100% pass
4. Deloitte QA Team Functional Tests Scripts written
5. Deloitte QA Team Feature/System testing executed for new configuration and code – 100% pass
6. Passes PO initial review and acceptance criteria outlined as part of the User Story is met

In order to accept the user story as Done by the Sprint Team, it must meet the Definition of Done criteria before it can go to Sprint Review.
Appendix
Sprint management overview

Sprints are managed via tracking of sprint burn down and the work plan is updated with % complete at the end of every sprint.

Actual Team velocity is tracked and used as an input to future sprint planning.
Sprint management – Tracking burn down

- Provides a visual representation of earned value over time
- Gives an earlier view into potential delivery issues
- Shows the Teams’ overall progress against the sprint goals (vs % complete progress of individual deliverables)
- Is updated daily (due to the short nature of sprints) and kept visible to the entire Team
- Supports the Team working together to complete the work and reduces silos
- Requires planned effort (from PE&PS or original estimate) and remaining effort (remaining work) for each work item in the sprint
- Once the sprint is complete, the project work plan will be updated with 100% complete for all completed and accepted sprint deliverables. The completed “planned effort” represents the Team’s actual velocity and will be used to update future sprint plans (if needed).